Available online at www.sciencedirect.com
SCIENCE DIRECT®
Graphical Models 67 (2005) 43–71
Graphical Models
www.elsevier.com/locate/gmod

Dynamic PDE-based surface design using geometric and physical constraints

Haixia Du*, Hong Qin

Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794-4400, USA

Received 19 May 2003; received in revised form 2 January 2004; accepted 8 June 2004
Available online 12 August 2004

Abstract

PDE surfaces, which are defined as solutions of partial differential equations (PDEs), offer many modeling advantages in surface blending, free-form surface modeling, and specifying surface's aesthetic or functional requirements. Despite the earlier advances of PDE surfaces, previous PDE-based techniques exhibit certain difficulties such as lack of interactive sculpting capabilities and restrained topological structure of modeled objects. This paper presents an integrated approach that can incorporate PDE surfaces into the powerful physics-based modeling framework, to realize the full potential of PDE methodology. We have developed a prototype system that allows interactive design of flexible topological surfaces as PDE surfaces and displacements using generalized boundary conditions as well as a variety of geometric and physical constraints, hence supporting various interactive techniques beyond the conventional boundary control. The system offers a set of sculpting toolkits that allow users to interactively modify arbitrary points, curve spans, and/or regions of interest across the entire PDE surfaces and displacements in an intuitive and physically meaningful way. To achieve real-time performance, we employ several simple, yet efficient numerical techniques, including the finite-difference discretization, the multigrid-like subdivision, and the mass-spring approximation of elastic PDE surfaces and displacements. In addition, we present the standard bivariant B-spline finite element approximations of dynamic PDEs, which can subsequently be sculpted and deformed directly in real-time subject to the intrinsic PDE constraints. Our experiments

* Corresponding author. Fax: 1-631-632-8834.
E-mail addresses: dhaixia@cs.sunysb.edu (H. Du), qin@cs.sunysb.edu (H. Qin).

1524-0703/$ - see front matter
doi:10.1016/j.gmod.2004.06.002

demonstrate many attractive advantages of the physics-based PDE formulation such as intuitive control, real-time feedback, and usability to both professional and common users.

Keywords: PDE surfaces; Geometric design; Deformable models; Interactive techniques; Physics-based modeling

1. Introduction and motivation

Surface modeling techniques are fundamental for many visual computing applications including interactive graphics, CAD/CAM, animation, and virtual environments. Frequently-used representation schemes for free-form surface modeling such as spline-based techniques [1–6] make use of simple polynomial functions in association with control points (and weights or knots). Spline-based models offer a unified mathematical formulation for free-form curves, surfaces, and solids. Despite the rapid advances in theoretical foundations and mathematical properties of free-form splines during the past several decades, traditional spline-based modeling techniques can be difficult, time-consuming, less natural, and counter-intuitive. This is primarily because free-form splines are frequently associated with tedious and indirect shape manipulations through time-consuming operations on a large number of (oftentimes irregular) control vertices, non-unity weights, and/or non-uniform knots. In addition, mathematical splines only model geometric attributes and require strong mathematical sophistication from users.

In sharp contrast, PDE techniques have various applications in graphics and modeling, such as surface fairing [7], model reconstruction [8,9], and shape metamorphosis [10], etc. Furthermore, PDE techniques provide an alternative way to model parametric surfaces [11–14]. PDE surfaces permit geometric objects to be defined and governed by a set of partial differential equations. In comparison with traditional control-point-based techniques, PDE surfaces offer many advantages:

- Natural physical processes are frequently characterized by PDEs. In principle, PDE surfaces are controlled by physical laws, so they are natural and close to the real world. They can potentially integrate geometric attributes with functional constraints for surface modeling, design, and analysis.
- The formulation of differential equations is well-conditioned and technically sound. Smooth surfaces with high-order continuity requirements can be readily defined through PDEs.
- Smooth surfaces that minimize certain energy functionals are oftentimes associated with differential equations according to the minimum principle of variational calculus. Hence, optimization techniques can be unified with PDE surfaces.
- Many powerful numerical techniques to solve PDEs are commercially available. Parallel algorithms can be deployed for large-scale problems in industrial settings.
- Users can easily understand the underlying physical process associated with PDEs, therefore, high-level intuitive and natural control is possible through the modification of physical parameters.

- PDE surfaces may potentially unify both geometric and physical aspects. They are invaluable throughout the entire modeling, design, analysis, and manufacturing tasks. Various heterogeneous requirements can be enforced and satisfied simultaneously.

Despite the rapid advances and modeling successes of PDE surfaces, they demand a lot of novel interactive techniques to realize their full potential. Typical modeling difficulties associated with PDE surfaces include:

- The prior work on PDE surfaces mainly concentrates on elliptic PDEs and is lack of interactive techniques for direct shape manipulation.
- Besides simple geometric conditions along PDE surface boundaries, as well as manual editing on PDE coefficients, there are few techniques for the direct manipulation of PDE surfaces in general.
- Traditional elliptic PDE surfaces only result from Hermite-like boundary conditions (i.e., boundary curves and their corresponding derivatives up to order n). More flexible and general boundary constraints have not yet been addressed.
- Conventional PDE techniques govern the entire parametric domain, but are unable to support localized geometric operations. Global control is less intuitive to manipulate.

To ameliorate it, we [15] proposed an interactive method and developed novel modeling techniques that can facilitate the direct manipulation and interactive sculpting of PDE surfaces. Our algorithms and design framework are founded upon the integrated principle of differential equations and physics-based modeling. To further promote the applicability of PDE surfaces in interactive graphics, CAD/CAM, and virtual engineering, we [16] extended both the geometric coverage and topological variation of PDE surfaces. Our new system provides users a set of more powerful sculpting tools than previously developed point-based editing capabilities. These toolkits allow PDE surfaces to be defined through the use of general, flexible boundary constraints. PDE surfaces of complicated geometry and diverse types of topology are readily available in our modeling environment. Other typical design tools in our environment include merging multiple surfaces, manipulations of isoparametric curves and/or arbitrary curve networks, editing any user-specified sub-surface, etc. Using our system, users are able to enforce both physical requirements and geometric criteria on PDE surfaces simultaneously with ease.

In this paper, we summarize the interactive techniques and the prototype software system facilitating the direct manipulation and interactive sculpting of PDE surfaces with flexible boundary conditions. To further extend PDE techniques for manipulation of existing models, we employ the PDE formulation to model displacements on parametric surfaces, which defines the surfaces as the original surfaces plus displacements and surface deformation can be achieved by manipulating the PDE governed displacements through the interactive toolkits. This extension allows users to directly model existing parametric surfaces through our system. It facilitates data exchange of PDE techniques with other parametric modeling methods. Our modeling algo-

rithms and design framework are founded upon the integrated methodology of physics-based modeling and differential equations, which provides real-time sculpting of PDE surfaces.

The remainder of this paper is structured as follows. Section 2 reviews the prior work of PDE surfaces, physics-based modeling techniques, and other interactive surface modeling techniques. In Section 3, we detail the PDE and physics-based modeling formulations and present our integrated approach. Section 4 presents novel techniques of directly manipulating PDE surfaces with generalized boundary constraints and flexible topology and PDE-based surface displacements model. We outline the system implementation and present our experimental results in Section 5. Finally, Section 6 gives the conclusion.

2. Background review

Since earlier 1970's, we have witnessed the ever-increasing popularity of the spline-based surface modeling techniques [2] in CAD/CAM applications. Non-uniform rational B-splines (NURBS), developed in 1975 and incorporated into initial graphics exchanged specification (IGES) in 1983, have become an industrial standard for modeling and data exchange in CAD/CAM. Various NURBS-based techniques have been developed during the past 20 years. Piegl and Tiller [6] detailed various NURBS-based modeling techniques such as interpolation/approximation of a set of data points and surface definition from a set of cross-sectional curves.

In contrast to spline surfaces, Bloor and Wilson in 1989 [11,12] introduced a different method—PDE surfaces—which defines smooth surfaces as solutions of an *elliptic* PDE. Since the initial application on surface blending, PDE surfaces have broadened their applications in surface description, functional design, solid modeling, and B-spline approximation in recent years. In principle, a PDE surface is governed by a set of boundary-value conditions and global coefficients associated with the elliptic PDE. This method has the advantage that most of the information defining a surface can be derived from its boundaries, which permits a surface to be generated and controlled through a relatively small set of parameters compared with the spline-based methods and subdivision schemes. This PDE technique can be used to generate piecewise free-form surfaces [13]. By varying the boundary conditions and control coefficients in the PDE, designers can obtain various surface shapes. Furthermore, Lowe et al. [14] presented a method with which certain engineering design criteria such as functional constraints can be incorporated into the geometric design of PDE surfaces. Therefore, it is possible to simultaneously introduce geometric constraints, aesthetic criteria, and physical and engineering requirements into the design process. Additionally, Bloor and Wilson [17] developed an algorithm that approximates PDE surfaces using standard B-splines. This work intends to demonstrate that PDE surfaces are virtually compatible with other mature and well established techniques based on popular splines for surface design. Hence, PDE surfaces can be readily incorporated into existing commercial design systems. Later on, in 1993,

PDE solids were formulated in terms of parametric boundary surfaces by Bloor and Wilson [18], which further expands the geometric coverage of PDE methodology. In [18], it shows the specific PDE can be used to facilitate the accurate analysis of mass attributes or physical properties for PDE objects. For certain simple boundary conditions, the elliptic PDEs can be solved analytically, i.e., PDE surfaces in these cases have a close-form formulation that frequently involves functions of Fourier series. However, for general boundary conditions, a PDE solution will have to be sought numerically instead. Later on, Bloor and Wilson [19] derived a set of approximate analytic solutions for PDEs in close-form for general boundary conditions. In 1999, Ugail et al. [20] have developed some techniques for interactively defining and changing boundary conditions to construct PDE surfaces.

However, the above methods and techniques can only afford users indirect and non-intuitive shape manipulation on PDE surfaces. Physics-based modeling, in contrast, offers users a means to overcome the drawback of indirect design mechanism associated with PDE surfaces. It is possible to unify physics-based modeling methodology with PDE approach, mainly because the dynamic behavior of physics-based models is also controlled by differential equations (e.g., Lagrangian equations of motion). Hence, physics-based modeling augments (rather than replaces) the existing PDE methodology, offering extra advantages for shape modeling. Free-form deformable models were initially introduced to computer graphics by Terzopoulos et al. [21] in 1987. Terzopoulos and Fleischer [22,23] demonstrated simple interactive sculpting using viscoelastic and plastic models. Celniker and Gossard [24] developed an interesting prototype system for interactive free-form design based on the finite-element optimization of energy functions proposed in [23]. Terzopoulos and Qin [25,26] formulated a novel model for interactive sculpting using Dynamic NURBS (D-NURBS). Ye et al. [27] incorporated certain functional constraints into the design process of geometric shapes. Dachille et al. [28] presented a haptic approach for the direct manipulation of physics-based B-spline surfaces. In general, physics-based techniques have various applications in visual computing areas [29–34]. Since the majority of physical phenomena can be characterized by differential equations, it is necessary to bridge the gap between geometric PDE surfaces and physics-based modeling approaches towards the realization of the full potential of PDE methods.

Some other interactive manipulation techniques for geometric surfaces are also available. Hsu et al. introduced direct manipulation of B-spline Free-Form Deformation (FFD) which used least squares to calculate the control points of B-spline FFD based on the direct movement of objects. Singh and Fiume [35] presented a geometric deformation technique called "wires" to deform objects using space curves and implicit functions. Biermann et al. [36] introduced a set of improved rules for subdivision models to generate smooth surfaces with normal control. Such techniques provide shape deformation based on control lattices or reference curve networks instead of working on the geometric objects directly. Furthermore, physical properties such as mass, damping, and stiffness attributes have not yet been considered for more realistic sculpting of objects in the above references.

3. Dynamic formulation and integration

This section formulates PDE surfaces and displacements, and discusses properties of the unified principle of PDE surfaces and physics-based modeling.

3.1. Elliptic PDEs

Throughout this paper, we focus on the fourth-order elliptic PDE which is a generalized version from the equation introduced by Bloor and Wilson [12]:

$$\left(\frac{\partial^2}{\partial u^2} + a^2(u,v)\frac{\partial^2}{\partial v^2}\right)^2 \mathbf{P}(u,v) = \mathbf{0}, \tag{1}$$

where u, v are parametric coordinates on the 2D parametric domain, $a(u,v)$ is a smoothing function of u and v that controls the behavior of PDE surfaces locally, and $\mathbf{P}(u,v) = [x(u,v)\ y(u,v)\ z(u,v)]^{\mathrm{T}}$ denotes the PDE surface in 3D space. Note that, in [12], the control coefficient is a constant a. To offer users more flexibility for interactive manipulation, we replace the constant coefficient using an arbitrary function of u and v, which can be defined by users. Because $a(u,v)$ may vary across $\mathbf{P}(u,v)$, local control on PDE surfaces can be achieved. Furthermore, although our system is focusing on this particular elliptic PDE, our mathematical derivation and its associated numerical techniques can be readily generalized to other PDEs. To solve (1), at least four boundary conditions are required to derive a unique solution. We further assume that a PDE surface is geometrically either closed or open along its two parametric directions (i.e., u and v). Therefore, our PDE surfaces may be topologically flexible, yielding diverse types of surfaces equivalent to four-sided open patches, spheres, cylinders, and tori. We restrain u and v to vary between 0 and 1, because reparameterization process can be conducted without changing the geometry of PDE surfaces if either u or v belongs to any $[a,b]$. Various boundary conditions can be imposed. To simplify our implementation, we classify PDE surfaces into three types: (1) open along both u- and v-directions, (2) open along u-direction and closed along v-direction, and (3) closed along both directions.

The four boundary curves in previous work are Hermite-like boundary conditions which comprise two curves defining a pair of the curved surface boundaries at the opposite side along one parametric direction (e.g., u-direction), and a pair of their associated derivative curves defining gradient information across the two curved boundaries. They are of the following form:

$$\begin{aligned} &\mathbf{P}(0,v) = \mathbf{c}_0(v), \quad \mathbf{P}(1,v) = \mathbf{c}_1(v), \\ &\frac{\partial}{\partial u}\mathbf{P}(0,v) = \mathbf{d}_0(v), \quad \frac{\partial}{\partial u}\mathbf{P}(1,v) = \mathbf{d}_1(v). \end{aligned} \tag{2}$$

The boundary conditions of $\mathbf{P}(0,v)$ and $\mathbf{P}(1,v)$ define two boundary edges of the surface represented by (1). The derivative conditions at $u = 0$ and $u = 1$ in (2) determine surface normal at the corresponding surface boundaries. These derivatives significantly influence the overall shape of the underlying PDE surface.

In our system, we generalize the boundary conditions of a PDE surface to a curve network. This can enhance the cross-sectional design of PDE surfaces from a set of curves. For instance, consider the design techniques of Gordon surface and Coons patch, our generalized boundary constraints can have the following form

$$\mathbf{P}(u_i, v) = \mathbf{f}_i(v), \quad \mathbf{P}(u, v_j) = \mathbf{g}_j(u), \tag{3}$$

where $0 \leqslant u_i \leqslant 1$ and $0 \leqslant v_j \leqslant 1$, and $\mathbf{f}_i(v)$ and $\mathbf{g}_i(u)$ are isoparametric curves. Moreover, a set of non-isoparametric curves can be easily added into our formulation.

By interactively modifying generalized boundary constraints, users are capable of manipulating the entire surface in an *indirect* manner. This property offers the designer an efficient way to edit the PDE surface through a fewer number of parameters that define boundary curves.

3.2. Displacement model

The idea of displacing a surface by a function was introduced by Cook [37]. Displacement maps are often used for texture mapping of bumped surfaces or modeling of complicated detailed meshes of arbitrary topology with regular surface patches. The complex surface can be represented as a scalar/vector-valued displacement over smooth domain surface. The displacement maps can be viewed as images, and this type of representation facilitates the use of image processing operators for manipulating the geometric detail of an object. They are also compatible with modern photo-realistic rendering system [38]. The idea is also used in subdivision techniques to produce *displaced subdivision surfaces* [39] and multiresolution surfaces [40]. The displacement maps can decrease the complexity of the model. The advantages of this representation lie in its simplicity and flexibility. The natural hierarchical division between coarse and fine features allows rapid computation of local surface features, and makes the data structure ideal for rapid collision detection for interactive operations. Furthermore, if local features are represented by an array of scalar values, limited editing of the local geometry can be done rapidly by modifying the values in the displacement map [41]. An illustration of the idea of displacement models is shown in Fig. 1.

To further explore the modeling potentials of PDE techniques on existing surface models, we employ the PDE formulation on surface displacements, i.e., offsets of the

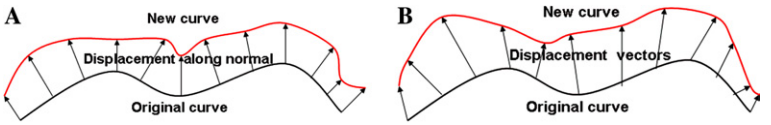

Fig. 1. Illustration of displacement models: (A) displacement curve model based on the displacements along curve normal; (B) displacement curve model based on the displacement vectors on the original curve.

input surface. The target surface will be the result by adding corresponding displacements onto the original surface. Different with popular displacement techniques, our PDE method will model displacement vector maps of the surface instead of scalar-valued maps associated with surface normal and leave the underlying surface untouched. The formulation of PDE displacement models is a slightly modified version of (1):

$$\begin{aligned} &\mathbf{P}(u,v) = \mathbf{P}^0(u,v) + \mathbf{O}(u,v), \\ &\left(\frac{\partial^2}{\partial u^2} + a^2(u,v)\frac{\partial^2}{\partial v^2}\right)^2 \mathbf{O}(u,v) = \mathbf{0}, \end{aligned} \tag{4}$$

where $\mathbf{P}(u,v)$ is the target surface, $\mathbf{P}^0(u,v)$ is the original surface, and $\mathbf{O}(u,v)$ is the corresponding surface displacements. Note that, (1) is the simplest case of (4) by simplifying the original surface to $\mathbf{P}^0(u,v) = \mathbf{0}$, i.e. shrinking to a point at the coordinate origin.

3.3. Numerical approximation techniques

Prior work on PDE surfaces mainly seeks close-form analytic solutions (e.g., Fourier series functions) to exploit many attractive properties associated with analytic formulations for surface design. However, in the interest of allowing arbitrary boundary conditions, we resort to numerical techniques that guarantee approximated solutions of the integrated formulation for PDE surfaces of flexible topology. Numerical algorithms also facilitate the material modeling of anisotropic distribution and its realistic physical simulation, where there are no close-form analytic solutions available for PDE surfaces. Among many mature techniques, we employ two popular numerical approaches to demonstrate the universal applicability of our framework: (1) finite-difference discretization, and (2) finite-element method based on B-spline approximation.

The finite-difference method is to transform a PDE to a system of algebraic equations by sampling the parametric domain into regular grids, then replacing all the partial derivatives in the differential equation with their discretized approximations on the sample points. The algebraic equations can then be solved numerically either through an iterative process or a direct procedure to obtain an approximated discrete solution to the continuous PDE.

Based on the Taylor series expansion of a continuous function $f(x)$, we derive the central-difference approximation of derivatives: $f'(x) \approx (f(x+h) - f(x-h))/2h$, and $f''(x) \approx (f(x+h) - 2f(x) + f(x-h))/h^2$, where h is the space interval along x. This method can be generalized to compute partial derivatives for bivariate surface geometry, by dividing the continuous parametric domain of u and v into m and n discretized points, respectively. This allows us to discretize $\mathbf{P}(u,v)$ as a collection of data points $\mathbf{p}_{i,j}$. For example, given a pair of parametric value $[u_i, v_j]$, $\mathbf{P}(u_i, v_j)$ becomes $\mathbf{p}_{i,j}$, $\mathbf{P}(u_i + \Delta u, v_j)$ becomes $\mathbf{p}_{i+1,j}$, and $\mathbf{P}(u_i - \Delta u, v_j)$ becomes $\mathbf{p}_{i-1,j}$, etc. (Fig. 2). The discretized partial derivatives have the following forms:

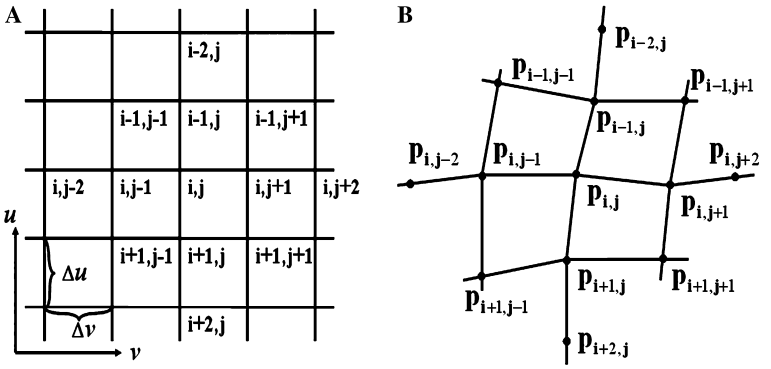

Fig. 2. Illustration of surface discretization: (A) discretization of the 2D parametric domain; (B) illustration of the point discretization of a continuous surface.

$$
\begin{aligned}
\frac{\partial^4}{\partial u^4}\mathbf{p}_{i,j} &\approx \frac{1}{\Delta u^4}[\mathbf{p}_{i+2,j} - 4\mathbf{p}_{i+1,j} + 6\mathbf{p}_{i,j} - 4\mathbf{p}_{i-1,j} + \mathbf{p}_{i-2,j}],\\
\frac{\partial^4}{\partial v^4}\mathbf{p}_{i,j} &\approx \frac{1}{\Delta v^4}[\mathbf{p}_{i,j+2} - 4\mathbf{p}_{i,j+1} + 6\mathbf{p}_{i,j} - 4\mathbf{p}_{i,j-1} + \mathbf{p}_{i,j-2}],\\
\frac{\partial^4}{\partial u^2\partial v^2}\mathbf{p}_{i,j} &\approx \frac{\mathbf{p}_{i+1,j+1} - 2\mathbf{p}_{i,j+1} + \mathbf{p}_{i-1,j+1} - 2\mathbf{p}_{i+1,j}}{\Delta u^2\Delta v^2}\\
&\quad + \frac{4\mathbf{p}_{i,j} - 2\mathbf{p}_{i-1,j+1} + \mathbf{p}_{i+1,j-1} - 2\mathbf{p}_{i-1,j} + \mathbf{p}_{i-1,j-1}}{\Delta u^2\Delta v^2},
\end{aligned}
\tag{5}
$$

where Δu and Δv denote the spatial intervals along u and v directions in the parametric domain, respectively.

Meanwhile, we discretize the blending bivariate function $a(u,v)$ into a set of $a_{i,j}$ that have a one-to-one correspondence with surface points $\mathbf{p}_{i,j}$. Therefore, applying the boundary constraints on the discretized boundary sample points and using (5) for the interior points, we obtain $m \times n$ difference equations for (1):

$$\mathbf{QP} = \mathbf{b}, \tag{6}$$

where $\mathbf{Q}$ contains the difference operators associated with control coefficients for each sample point,

$$\mathbf{Q} = \begin{bmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,m\times n} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,m\times n} \\ \cdots & \cdots & \cdots & \cdots \\ q_{m\times n,1} & q_{m\times n,2} & \cdots & q_{m\times n,m\times n} \end{bmatrix},$$

and

$$\mathbf{P} = [\,\mathbf{p}_{1,1}^{\top}\,\mathbf{p}_{1,2}^{\top}\cdots\mathbf{p}_{m,n}^{\top}\,]^{\top}, \quad \mathbf{b} = [\,\mathbf{b}_1^{\top}\,\mathbf{b}_2^{\top}\,\ldots\,\mathbf{b}_{m\times n}^{\top}\,]^{\top}.$$

Similarly, the PDE for displacements in (4) can be discretized as $\mathbf{QO} = \mathbf{b}$. Thus, (4) has the approximated numerical form

$$\begin{aligned} \mathbf{P} &= \mathbf{P}^0 + \mathbf{O} \\ \mathbf{QO} &= \mathbf{b}. \end{aligned} \tag{7}$$

Different topological types of PDE surfaces are available in our system. First, the surface can be closed along one parametric direction (e.g., v-direction), in which case the points on $v = 0$ are the same as those on $v = 1$. The central-difference scheme suffices for the computation of partial derivatives with respect to v. Second, the PDE surface is open along both u and v directions. In this case, the computation of partial derivatives on two boundary curves requires special care, and forward or backward difference approximation shall be utilized along the open boundary curves instead. Third, the PDE surface is closed along both directions, and the central-difference approximation can be applied anywhere across the surface geometry. Boundary constraints determine all the point coordinates lying on the user-specified curves. Moreover, for the Hermite-like boundary conditions, the initial derivative information across boundary curves determines additional point coordinates in the vicinity of specified boundaries (e.g., $\mathbf{p}_{2,j}$ and $\mathbf{p}_{m-1,j}$) that are adjacent to two boundary curves at $u = 0$ ($\mathbf{p}_{1,j}$) and $u = 1$ ($\mathbf{p}_{m,j}$). Arbitrary boundary conditions can be easily enforced without any difficulty using finite-difference method. Note that, in spite of certain combinations of constraint imposition shown in our experiments, in general this type of elliptic PDEs allows the boundary conditions to be explicitly formulated in arbitrary form. This permits designers to choose (various) constraints based on diverse design tasks. The same flexible topological feature also applies to PDE displacements. However, the topological type of displacements depends on the underlying original surface. This gives designers more freedom when modeling surfaces of flexible topology.

3.4. Physics-based modeling

To obtain interactive and direct sculpting on the PDE surfaces, we integrate the physics-based modeling techniques with the PDE model.

An elastic deformable model is characterized by the position $\mathbf{p}(u,v,t)$, velocity $\dot{\mathbf{p}}(u,v,t)$ (which stands for $\frac{\partial \mathbf{p}(u,v,t)}{\partial t}$), and acceleration $\ddot{\mathbf{p}}(u,v,t)$ (i.e., $\frac{\partial^2 \mathbf{p}(u,v,t)}{\partial t^2}$) along with material properties such as mass, damping, and stiffness distributions. These values are defined over the surface as functions $\mu(u,v)$,$\gamma(u,v)$, and $\rho(u,v)$, respectively, which oftentimes can be considered to be constant at certain time. However, these material quantities are allowed to be modified by users interactively and directly over the surface model. In general, to simulate the real-time performance of direct manipulations, a continuous dynamic surface can be discretized into a collection of mass-points connected by a network of springs across nearest neighbors (and/or along both diagonals) in the parametric domain. Other springs can be incorporated into the discretized surface if certain types of dynamic behavior are more desirable. We use a mass-spring model because of its simplicity and the critical need of real-time surface sculpting.

By applying Lagrangian mechanics, we obtain a set of second-order differential equations that govern the physical behavior of the underlying physics-based mass-spring model:

$$\mathbf{M}\ddot{\mathbf{P}} + \mathbf{D}\dot{\mathbf{P}} + \mathbf{K}\mathbf{P} = \mathbf{f}, \tag{8}$$

where $\mathbf{M}$ is a mass matrix, $\mathbf{D}$ is a damping matrix, $\mathbf{K}$ is a stiffness matrix, $\mathbf{P}$ denotes the collection of the sample points of the model, and $\mathbf{f}$ represents the entire external force applied on the surface. The force at every mass-point in the mesh is the sum of all possible external forces: $\mathbf{f} = \sum \mathbf{f}_{\text{ext}}$. The internal forces are generated by the connecting springs, where each spring is modeled with force $\mathbf{f}_{\text{int}} = k(\mathbf{l} - \mathbf{l}_0)$ according to Hook's law. The rest length $\mathbf{l}_0$ of each spring is determined during the initialization of the PDE surface. Our system allows users to modify the rest length interactively.

It now sets a stage for us to combine the benefits of both physics-based models and PDE surfaces to support the interactive surface design with real-time feedback. By attaching mass points to the geometric grid and adding springs between adjacent points on the discretized PDE mesh, as shown in Fig. 3, we can integrate the Lagrangian mechanics with (6) to form the unified framework, then we obtain a dynamic version of PDE surfaces:

$$\mathbf{M}\ddot{\mathbf{P}} + \mathbf{D}\dot{\mathbf{P}} + (\mathbf{K} + \mathbf{Q})\mathbf{P} = \mathbf{b} + \mathbf{f}, \tag{9}$$

where both the velocity and the acceleration of $\mathbf{P}$ can be discretized along the time axis analogously:

$$\ddot{\mathbf{P}} \approx (\mathbf{P}^{t+\Delta t} - 2\mathbf{P}^{t} + \mathbf{P}^{t-\Delta t})/\Delta t^2, \quad \dot{\mathbf{P}} \approx (\mathbf{P}^{t+\Delta t} - \mathbf{P}^{t-\Delta t})/2\Delta t.$$

The composite dynamic PDE displacement surface model can be obtained in the same way:

$$\begin{aligned} &\mathbf{P} = \mathbf{P}^0 + \mathbf{O}, \\ &\mathbf{M}\ddot{\mathbf{O}} + \mathbf{D}\dot{\mathbf{O}} + (\mathbf{K} + \mathbf{Q})\mathbf{O} = \mathbf{b} + \mathbf{f}, \end{aligned} \tag{10}$$

where

$$\ddot{\mathbf{O}} \approx (\mathbf{O}^{t+\Delta t} - 2\mathbf{O}^{t} + \mathbf{O}^{t-\Delta t})/\Delta t^2, \quad \dot{\mathbf{O}} \approx (\mathbf{O}^{t+\Delta t} - \mathbf{O}^{t-\Delta t})/2\Delta t.$$

By allowing the PDE model to dynamically deform in time domain, users will have a natural feeling when they interactively manipulate the PDE model, which is lacking without Lagrangian equations of motion. Furthermore, material properties can be introduced to govern the behavior of the underlying PDE model. This *hybrid* formulation permits users to obtain a surface that satisfies both geometric criteria and functional requirements at the same time.

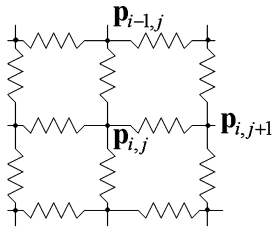

Fig. 3. Mass-spring network in the vicinity of surface point $\mathbf{p}_{i,j}$.

In this paper, we conduct various experiments on different numerical techniques in solving (6), (7), (9), and (10) to compare their performance. In general, we either resort to direct approaches or make use of iterative techniques. Certain variants of iterative techniques exist for solving the above difference equations [42]. We solve the above equations using Gauss–Seidel iteration, which starts from the initial guess (approximated values) of the discretized surface points, then recursively calculates the data points in a pre-defined order. After a finite number of iteration steps, the value obtained through the recursive approach are considered to be extremely close to the accurate solution. To further speed up the convergent rate of Gauss–Seidel iteration, we take into account the error factor that is defined by the difference between the approximation and the real value. This leads to the method of successive over-relaxation iteration, or SOR iteration. With SOR iteration, an *Over-Relaxation factor* $\sigma(1 < \sigma \leqslant 2)$ is introduced to achieve a better approximation at each step. When $\sigma = 1$, it reduces to Gauss–Seidel iteration. The different choice of σ leads to different convergent speed, and the optimal value of σ is allowed to vary depending on different problems. Our system provides the interactive modification of the value of σ, which is suitable for the solution of different kinds of linear equations. When using iterative approaches to solve the difference equations of PDEs, the initial guess plays a significant role that affects the convergent speed. Hence, extra cares need to be taken to warrant fewer calculations and better time performance. Furthermore, we take advantage of the multi-grid-like subdivision method to speed up the numerical integration. The surface is first solved on the coarsest sample points, and then it is refined into a finer grid whose initial values are computed through the simple linear interpolation or more complicated subdivision schemes such as Butterfly subdivision [43] or quadrilateral interpolating subdivision [44]. As a result, the convergent rate of our multi-grid technique can be greatly increased. In addition, this method allows the user to control the error bound of the approximated solution.

4. Interactive surface manipulation

This section details various interactive techniques and explains the implementation algorithm for PDE surface editing.

4.1. PDE surface initialization

Our system supports three topological types of PDE surfaces. At the beginning of the initialization phase, the user must specify the surface type, i.e., whether the surface is open or closed along u and v directions. Because any direct manipulation must be based on the user-defined initial surface, we need to select boundary conditions to generate a PDE surface as an initial step. We provide users two different ways to set up the boundary conditions of the PDE surface. First, users can interactively input some control points by clicking/dragging the mouse at desired locations on the screen, and the system will calculate cubic B-spline curves as boundary curves,

boundary derivative curves, and other special curves the PDE surface must interpolate. The boundary derivative curves are curves corresponding to boundary curves, respectively. The difference between any point on a boundary curve and its correspondence on the associated derivative curve will be used to determine both magnitude and direction of the tangent vector across the boundary curve. Alternatively, users are allowed to define the boundary conditions using certain analytic functions. The point coordinates are sampled along the analytic curves, and are saved into a data file. The system then can access data files and initialize the PDE surface based on these curves. After boundary conditions are determined, the PDE surface can be derived from the solution of the linear equations subject to these conditions. If boundary curves are B-splines, we can modify their shape by changing B-spline control points. Subsequently, the entire PDE surface will be re-computed and modified with the new boundary conditions. If boundary curves are obtained through certain analytic functions, we can calculate their B-spline approximations and modify their shapes.

4.2. Generalized boundary constraints

The solution of (1) is subject to boundary conditions. In general, there are several types of boundary conditions according to the information they contain. In this paper, we consider three types of boundary conditions: (1) Hermite-like conditions; (2) Coons-like conditions; and (3) Gordon-like conditions in analogy with their corresponding free-form surface formulation.

Hermite-like conditions include positions and the first-order or even higher-order derivatives of boundary curves. For the fourth-order PDE shown in (1), the boundary conditions may be Hermite-like (i.e., two boundary curves at $u = 0$ and $u = 1$, and their corresponding first-order derivatives). Boundary curves define the edges of the surface and corresponding derivative curves determine the gradient information across the boundaries, which outline the surface shape. Figs. 7 and 10 show examples of this type of conditions.

For any four-sided surface patch, there are four boundary curves in general. In our case, the boundary curves are those at $u = 0$, $u = 1$, $v = 0$, and $v = 1$, respectively. Such constraints, in analogy with Coons patch, are considered as Coons-like boundary conditions. Using such conditions, we can easily obtain surfaces that are open along both u-direction and v-direction, or closed along v and open along u. Note that, for surfaces that are closed only along v, it is equivalent to consider that two boundary curves at $v = 0$ and $v = 1$ are the same. Fig. 8 has an example of this boundary type.

Although four boundary curves can define PDE surfaces, they are far from enough to formulate complicated geometry or shapes with details. In this case, we need to define a curve network containing features that the PDE surface must interpolate. This kind of boundary constraints is a direct generalization of Gordon surfaces [3]. Hence, the Gordon-like boundary conditions consist of a family of isoparametric curves $\mathbf{P}(u_i,v) = \mathbf{f}_i(v)$ and $\mathbf{P}(u,v_j) = \mathbf{g}_j(u)$, where $0 \leqslant u_i \leqslant 1$ and $0 \leqslant v_j \leqslant 1$. We show an example of this type of surface construction (Fig. 8).

4.3. Subdivision on the PDE surface

Although the iterative techniques are easily implemented, oftentimes the large number of sample points of a PDE surface result in the slow convergence of such techniques. To improve the computation performance, we develop a multi-grid approximation based on popular subdivision schemes. At first, we can start with a small number of sample points on the coarsest grid of a PDE surface, the coarse solution of the PDE surface can be easily obtained quickly. Second, users can refine the coarse mesh through subdivision and use the new subdivided mesh as an initial guess for successive iterations. The finer grid is then computed iteratively to achieve a more accurate and smoother solution of the PDE surface. During the multi-grid process, the up-sampling of all generalized boundary curves is achieved through the use of four-point interpolatory subdivision scheme [43] to guarantee the smoothness requirement of the refined curves. Given control points $\{\mathbf{p}_i^0\}_{i=-2}^{n+2}$, the points at level $k+1$ of the subdivision are defined by

$$\begin{aligned} \mathbf{p}_{2i}^{k+1} &= \mathbf{p}_i^k, \\ \mathbf{p}_{2i+1}^{k+1} &= \left(\frac{1}{2}+w\right)(\mathbf{p}_i^k + \mathbf{p}_{i+1}^k) - w(\mathbf{p}_{i-1}^k + \mathbf{p}_{i+2}^k), \end{aligned} \tag{11}$$

where $-1 \leqslant i \leqslant 2^k n + 1$.

According to [43], the curve is tightened toward the control polygon as $w \to 0$, and for any $0 < w < (\sqrt{5}-1)/8$, the interpolated curve is a C^1 curve. Because w influences the smoothness of the boundary curves, the system allows users to change the value of w to obtain satisfactory results. For refinement of surface points, we currently employ linear interpolation to obtain an initial guess and use the PDE to approximate the solution. However, several subdivision techniques can be used to obtain initial guesses of finer grids from coarse approximations. Fig. 9 shows an example using the multi-grid subdivision.

4.4. Manipulating boundary conditions

Because boundary curves are defined by B-spline curves, or have B-spline approximation, we can modify the shape of the PDE surface globally by changing B-spline control points of boundary curves. Fig. 10 shows an example.

4.5. Modifying control function $a(u,v)$

The blending coefficient function $a(u,v)$ can influence the surface shape. The value of $a(u,v)$ controls the relative smoothness and the level of variable dependence between the parametric directions of u and v. For a large $a_{i,j}$ at point $\mathbf{p}_{i,j}$, changes in the u-direction occur within a relatively short length scale, i.e., it is $1/a_{i,j}$ times the length scale in the v-direction in which similar changes can take place. Consequently, users can control how boundary conditions influence the interior of the surface by

modifying the length scale (i.e., $a_{i,j}$) at arbitrary point on the PDE surface. In general, $a(u,v)$ can be interactively *painted* over the entire surface (see Fig. 11).

4.6. Joining multiple surfaces

Oftentimes a single PDE surface may not satisfy complicated design requirements, because real-world objects exhibit both complex topological structure and irregular geometric shape. We can piece multiple PDE surfaces together for this purpose. In our system, users can join $n-1$ PDE surfaces sequentially by specifying $2n$ Hermite-like boundary conditions (where $n \geqslant 3$). Note that, $2n$ conditions are necessary to satisfy C^1 continuity, because two neighboring PDE patches share one common boundary and the tangent vectors across the shared boundary should be same. Note that, because our coefficient function $a(u,v)$ in (1) may vary throughout the $u-v$ domain, we can consider the technique of joining multiple surfaces to be equivalent to generating one *larger* PDE surface with local control. Fig. 7 has an example.

4.7. Sculpting tools and geometric constraints for global and local deformation

By changing the boundary curves, we can modify the entire shape of a PDE surface. However, when the global appearance of a PDE surface is satisfactory, any subsequent sculpting via boundary conditions may destroy certain existing features of the underlying surface. In this situation, making small changes on a localized region is more desirable. This can be done by enforcing additional constraints on the PDE surface. Note that, the original finite-difference formulation consists of $m \times n$ equations and $m \times n$ unknowns, i.e., the coefficient matrix is a square matrix. The introduction of additional conditions forces the system to incorporate a set of new equations into the original set. There are two ways to solve such a system with those additional constraints. One way is to treat the constraints as hard constraints, i.e., the additional equations must be satisfied. This can be done by replacing the corresponding equations in the original system with these hard constraints. For example, if we want to move a sample point $\mathbf{p}_{i,j}$ on the discretized surface to a new location, $\mathbf{p}_0$, the equation $\mathbf{p}_{i,j} = \mathbf{p}_0$ will be used to replace the corresponding discretized difference equation approximating the PDE at the point $\mathbf{p}_{i,j}$, i.e., $q_{(i-1)\times n+j,(i-1)\times n+j} = 1$, all other $q_{(i-1)\times n+j,k} = 0$ for $k \neq (i-1)\times n + j$, and $\mathbf{b}_{(i-1)\times n+j} = \mathbf{p}_0$ in (6). This method works well if the additional constraints are of linear form (e.g., fixing a subset of certain unknowns or three points must be co-linear, etc.). As a result, (6) becomes

$$\mathbf{Q}_c\mathbf{P} = \mathbf{b}_c, \tag{12}$$

where $\mathbf{Q}_c$ and $\mathbf{b}_c$ are obtained by incorporating additional hard constraints. And it can be solved by iterative methods analogously.

Alternatively, these additional conditions can be treated as soft constraints by directly adding the equations of the constraints into the original linear system. Using the same example, $\mathbf{p}_{i,j} = \mathbf{p}_0$ will be added into (6) as the $m \times n + 1$ row, i.e.,

$q_{m\times n+1,(i-1)\times n+j}=1$, all other $q_{m\times n+1,k}=0$ for $k\neq(i-1)\times n+j$, and $\mathbf{b}_{m\times n+1}=\mathbf{p}_0$. By adding all the additional constraints, (6) will then turn into

$$\mathbf{Q}_\mathbf{a}\mathbf{P}=\mathbf{b}_\mathbf{a}, \tag{13}$$

where $\mathbf{Q}_\mathrm{a}$ has $m\times n+k$ rows and $m\times n$ columns with $k>0$ as the number of additional constraints. We now have an over-constrained system with more equations than the number of unknowns. This system cannot be directly solved through the aforementioned techniques. Instead, we solve the above equations in a least-square manner [42]. The least-square approximation is a solution of the following equation

$$\mathbf{Q}_\mathbf{a}^\top\mathbf{Q}_\mathbf{a}\mathbf{P}=\mathbf{Q}_\mathbf{a}^\top\mathbf{b}_\mathbf{a}. \tag{14}$$

Now the composite matrix becomes a square matrix, and the equations can be solved using aforementioned techniques.

Because the additional constraints discussed in this paper are all expressed as linear equations, we use the hard-constraint approach to solve the linear system. Note that, other more robust algorithms such as singular value decomposition are amenable to our PDE sculpting as well.

4.7.1. Manipulating surface points

To manipulate a surface directly, one desirable way is to enforce the PDE surface interpolating a specific location in 3D space. This can be done by picking a point on the sampling surface grid, e.g., $\mathbf{p}_{i,j}$, then dragging it to the position where the surface should pass through. Then the PDE is re-solved, with this additional interpolating condition for $\mathbf{p}_{i,j}$ associated with the original difference equation system. Users can edit a set of points in a sequential order, and the modified surface interpolates all the selected data points. Fig. 12A shows a modified PDE surface by changing the position of one point on the original surface.

4.7.2. Changing surface normal

We can also manipulate the surface normal on any sample point to obtain the local editing capability on the surface in the vicinity of the selected point, as demonstrated in Fig. 4. This is because the normal of a continuous surface at a selected point can be approximated by the neighboring points using finite-difference method:

$$\mathbf{n}_{i,j}=\frac{\mathbf{p}_{i+1,j}-\mathbf{p}_{i-1,j}}{2\Delta u}\times\frac{\mathbf{p}_{i,j+1}-\mathbf{p}_{i,j-1}}{2\Delta v}.$$

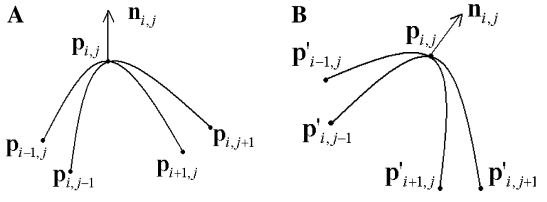

Fig. 4. The change of the normal of a point corresponds to the change of positions of neighbor points: (A) normal $\mathbf{n}_{i,j}$ of $\mathbf{p}_{i,j}$; (B) the new normal $\mathbf{n}_{i,j}$ leads to the change of points $\mathbf{p}_{i-1,j}$, $\mathbf{p}_{i+1,j}$, $\mathbf{p}_{i,j-1}$, and $\mathbf{p}_{i,j+1}$.

When we change the surface normal at a selected point, our system will subsequently compute its four neighboring points according to the new normal direction. In our implementation, we simply enforce four new equations in the difference equation system (6). By solving the constrained system, we obtain a modified surface with the normal constraint at the selected point. An example for this kind of constraints is shown in Fig. 12B.

4.7.3. Editing surface curvature

Since the curvature measures the intrinsic shape of a curve/surface, we can change the shape of a PDE surface by modifying the curvature at arbitrary point. The curvature of a curve can be defined by: $\kappa = \frac{\|\mathbf{x}' \times \mathbf{x}''\|}{\|\mathbf{x}'\|^3}$. Now we consider the curvature at any surface point along u-direction and v-direction, respectively:

$$\kappa_u = \frac{\left\|\frac{\partial \mathbf{p}}{\partial u} \times \frac{\partial^2 \mathbf{p}}{\partial u^2}\right\|}{\left\|\frac{\partial \mathbf{p}}{\partial u}\right\|^3}, \quad \kappa_v = \frac{\left\|\frac{\partial \mathbf{p}}{\partial v} \times \frac{\partial^2 \mathbf{p}}{\partial v^2}\right\|}{\left\|\frac{\partial \mathbf{p}}{\partial v}\right\|^3}.$$

These equations imply that changing curvature will modify the positions of the neighboring points. But if we solve the above equations directly, we need to deal with non-linear equations. To avoid this and keep the implementation algorithms simple for real-time geometric design, we approximate the solution of curvature modification by moving the neighboring points along the corresponding parametric direction (refer to Fig. 5), so we interactively edit the curvature information by changing the distribution of the neighboring points (e.g., $\mathbf{p}_{i-1,j}$ and $\mathbf{p}_{i+1,j}$ for κ_u at $\mathbf{p}_{i,j}$). After we compute the new position of relevant neighbors corresponding to the curvature manipulation, we can incorporate these known values of data points into the system and re-solve the equations to derive a new surface that satisfies a set of curvature constraints simultaneously. In Fig. 12C, we modify a PDE surface with curvature constraints.

4.7.4. Curve constraints

Although point-based conditions provide designers useful manipulation tools, point editing is less appropriate when users are faced with complicate design requirements. We provide editing tools that afford the intuitive specification of curve-based constraints. First, users can select a source curve on the PDE surface by picking points on the $u - v$ domain. The curve is allowed to be of arbitrary form because the selected points may have arbitrary values of u and v, giving users more freedom

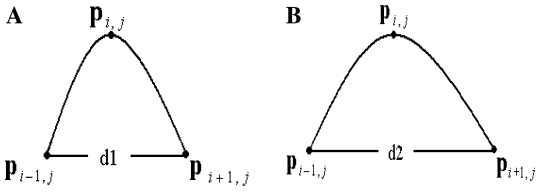

Fig. 5. Curvature modification via the change of the distance between the neighboring sample points: (A) high curvature due to reducing distance; (B) low curvature due to stretching distance.

for the effective surface editing. Second, users may interactively specify a cubic B-spline curve as the destination curve which will then be mapped to the selected surface curve. The B-spline curve shares the same number of sample points as that of the source curve. Third, our system will enforce the source curve to be in the same shape as the destination curve. The mapping of B-spline destination curve adds a number of new linear equations into (6), and the PDE surface will be modified accordingly. Users can freely modify or even re-define a destination curve which leads to different surface geometry. In principle, boundary conditions can be special variants of curve-based constraints. Fig. 14 illustrates an example of non-isoparametric curve constraints.

4.7.5. Region manipulation

Certain surface models exhibit special features in specific regions, hence it is more desirable to develop region-based editing tools toward the ultimate goal of feature-based design. Analogous to the aforementioned curve tool, our system can map a user-specified B-spline patch onto a region of interest over the PDE surface. First, users select an area over the PDE surface and define a B-spline patch which are sampled to have the same number of grid points as those in the source region. Then our system maps the shape of the B-spline patch to the specified area. Users can interactively deform the B-spline patch or create a new destination patch that imposes area constraints on the PDE geometry (Fig. 15). Because the surface–surface mapping algorithm depends on the structure of sample grid, we only consider source regions with rectangular grid in the interest of simplicity.

4.7.6. Sculpting and trimming localized regions

Conventional PDE surfaces only support global manipulation, i.e., any modification results in a new surface undergone the global deformation. This deficiency severely restrains users' freedom of arbitrary surface manipulation at any localized region(s). To overcome this difficulty, we allow designers to freeze any specified area of a PDE surface which they do not want to change. This can be achieved in our system by marking a region in $u - v$ domain, then any changes outside this region will not affect any data points inside, or fixing the outside while modifying inside the region, as shown in Fig. 13. In addition, our system offers users functionalities to trim a PDE surface for arbitrary topological shapes. This can be done by removing material from the PDE surface either inside or outside the specified regions.

4.7.7. Interactive sculpting of PDE displacements

Our interactive sculpting toolkits can be applied to manipulate the vector-valued displacements controlled by the PDE formulation. Similar to the constrained PDE surface (12), the formulation for hard-constrained PDE surface displacements can be written as

$$\mathbf{Q_c O} = \mathbf{b_c}. \tag{15}$$

The surface deformation is done through the manipulation of displacements while the original surface remains untouched. This enhancement allows the system not

only to design the surfaces by boundary conditions but also to manipulate existing parametric models directly, which further broadens the applications of the PDE-based surface modeling system. Fig. 16 has examples of local point editing on the PDE displacements. Fig. 17 shows examples for curve manipulation and region sculpting of displacement models.

4.8. Sculpting dynamic PDE surfaces

Up to this point, we have discussed several methods to change the linear equation system and recalculate it to derive a new PDE surface satisfying various global and local geometric criteria such as position, normal, curvature, curve span, and region constraints. Because the time performance of standard numerical solvers depends on the number of the sample points on the PDE model, users oftentimes have to patiently wait for the final stable surface as the large number of equations are solved within the system. When the number of sample points is extremely large, the computation time is at the order of seconds/minutes. This significantly limits the interactivity of surface modeling and manipulation as no visual feedback between the initial and final states are provided. To ameliorate, we consider the integrated mass-spring PDE model whose dynamic behavior is governed by (9) or (10). The external force $\mathbf{f}$ can be computed implicitly based on various additional constraints. We then divide the time domain into small time steps and approximate both velocities and accelerations of data points through successive time intervals. We can dynamically manipulate the PDE surface with forces in real-time by solving

$$\mathbf{M}\ddot{\mathbf{P}} + \mathbf{D}\dot{\mathbf{P}} + (\mathbf{K} + \mathbf{Q}_c)\mathbf{P} = \mathbf{b}_c + \mathbf{f}. \tag{16}$$

The same idea can be applied to PDE displacement model for dynamic deformation and real-time manipulation.

Additional constraints that control the behavior of the PDE surface can result from the editing of material properties such as mass/damping quantities and stiffness distributions. When additional constraints are incorporated into our mass-spring model, the surface points gradually evolve along consecutive time steps, hence the number of iterations to solve (16) is very small (less than 10). This results in real-time performance.

4.9. B-spline approximation

To facilitate the data exchange capability of PDE surfaces with standard spline-based systems, we compute a B-spline approximation of the PDE surface throughout the manipulation process. A B-spline surface over $u - v$ domain can be defined as

$$\mathbf{P}(u, v) = \sum_{i=1}^{k} \sum_{j=1}^{l} B_{i,r}(u) B_{j,s}(v) \mathbf{c}_{i,j}, \tag{17}$$

where $B_{i,r}(u)$ and $B_{j,s}(v)$ are B-spline basic functions of u and v with the order r and s, respectively, $\mathbf{c}_{i,j}$ (where $1 \leqslant i \leqslant k$, and $1 \leqslant j \leqslant l$) are B-spline control points.

Oftentimes, the number of control points is less than the number of sample points on the PDE surface, therefore the B-spline approximation results in over-constrained linear equations whose unknowns are fewer than the number of equations. For example, given the $m \times n$ sample points on the PDE surface, the approximation using $k \times l$ control points results in the following formulation

$$\mathbf{BC} = \mathbf{P}, \tag{18}$$

where there are $m \times n$ linear equations with $k \times l$ unknowns. Assuming fixed parametrization of data points in B-spline approximation, the matrix $\mathbf{B}$ is a discretization of B-spline basis functions. Note that, $\mathbf{B}$ is a constant matrix. $\mathbf{C}$ is the collection of control points and $\mathbf{P}$ represents the collection of sample points on the PDE surface.

This over-constrained system for B-spline control points can be solved by multiplying $\mathbf{B}^{\mathrm{T}}$ on both sides of (18). Consequently, we obtain a B-spline surface that approximate the PDE surface by solving the B-spline control mesh in least-square sense, as shown in (19)

$$\mathbf{C} = (\mathbf{B}^{\top}\mathbf{B})^{-1}\mathbf{B}^{\top}\mathbf{P}. \tag{19}$$

Fig. 18 shows examples of B-spline approximation for PDE surfaces.

Meanwhile, we also use B-spline finite elements to approximate the dynamic model of PDE surfaces at each time step. This allows users to interactively manipulate the B-spline solution of PDE surfaces with forces in real-time. Because the B-spline control mesh is obtained using least-square fitting, the additional constraints of the approximated PDE surface are treated as soft constraints which result in a smoother solution for the PDE surface than the one obtained under hard constraints.

5. System implementation and results

This section outlines the system functionalities and presents our experimental results.

5.1. System functionalities

We have developed a prototype software system that permits users to interactively manipulate PDE surfaces and displacements with various constraints either locally or globally. The system is written in Visual C++ and runs on Windows systems. Fig. 6 illustrates the architecture of our prototype system. Our system provides the following functionalities:

5.1.1. Boundary conditions

Users can interactively input and edit several types of boundary conditions defined by cubic B-spline curves or commonly used analytic functions, and obtain PDE surfaces satisfying these constraints. Boundary conditions can also be modified freely as curve-based constraints. Moreover, the system offers a multi-grid-like subdivision scheme to improve the time performance of our system.

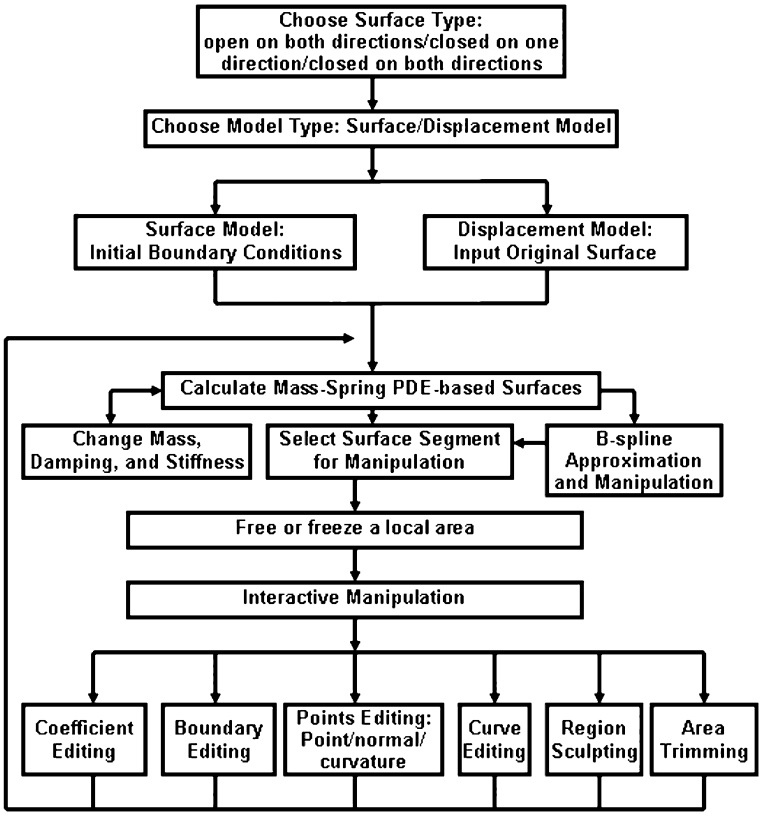

Fig. 6. System architecture.

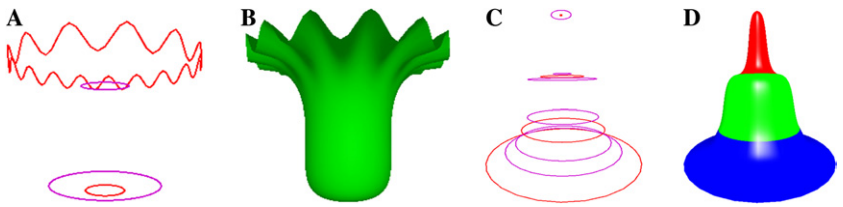

Fig. 7. The PDE surfaces from Hermite-like boundary conditions: (A) boundary conditions, where boundary curves are in red and derivative curves in purple; (B) the surface subject to (A); (C) three sets of boundary and derivative curves for connected PDE surface; (D) the connecting result. (For interpretation of the references to colours in this figure legend, the reader is referred to the web version of this paper.)

5.1.2. Displacement models

To additionally broaden the applications of PDE methods on surfaces, we extend the PDE formulation to model surface displacements, which represent the offsets of a given surface. This enables our PDE models to model a large set of surfaces of flexible topology.

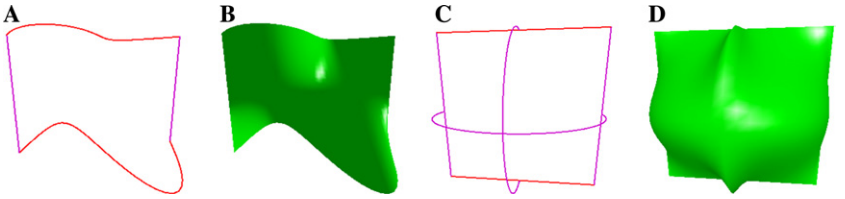

Fig. 8. The open PDE surfaces with Coons-like and Gordon-like boundary conditions: (A) Coons-like boundary curves; (B) the corresponding surface of (A); (C) Gordon-like curve network with curves at $u = 0$, $u = 0.5$, $u = 1$, $v = 0$, $v = 0.5$, and $v = 1$; and (D) the PDE surface from (C).

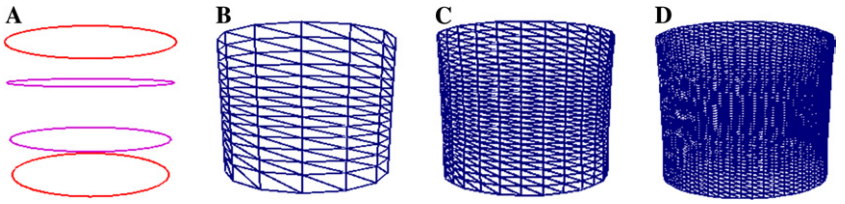

Fig. 9. The multigrid solution for a PDE surface: (A) initial boundary conditions; (B) PDE surface of sample grids 15×15; (C) PDE surface of sample grids 30×30; and (D) PDE surface of sample grids 60×60. Note that, w is 0.1 in this example.

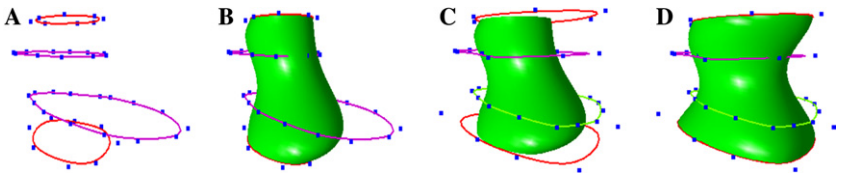

Fig. 10. Changing direct-input B-spline boundary conditions of a PDE surface: (A) initial B-spline boundary curves with control points; (B) the corresponding PDE surface; (C) modified boundary conditions; and (D) the modified surface.

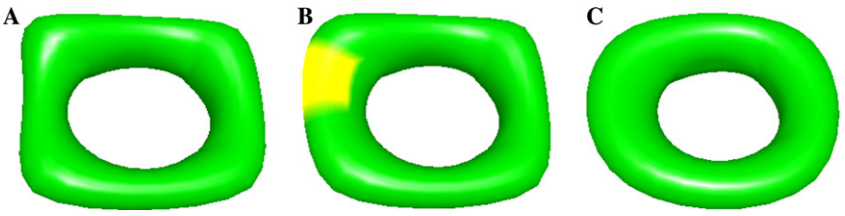

Fig. 11. Effects of changing blending coefficient $a(u, v)$: (A) the PDE surface of $a(u, v) = 3.0$; (B) the surface after changing value of $a(u, v)$ on the yellow part on the surface to 5.0; and (C) the surface by setting $a(u, v) = 5.0$ for all sample points.

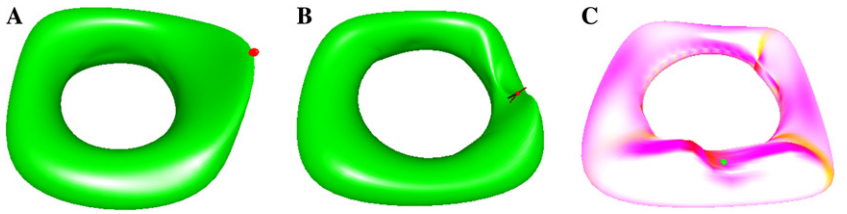

Fig. 12. Examples of point-based manipulation: (A) changing the location of a surface point; (B) normal modification of a selected point on the PDE surface; and (C) the modified surface after changing curvature at a selected point.

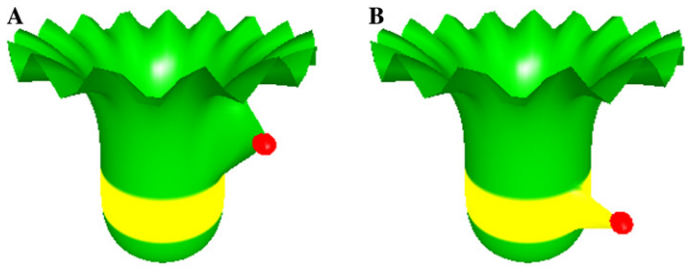

Fig. 13. Examples of surface manipulation with fixed regions: (A) the yellow part on the surface is fixed; (B) only yellow part is allowed to change. (For interpretation of the references to colours in this figure legend, the reader is referred to the web version of this paper.)

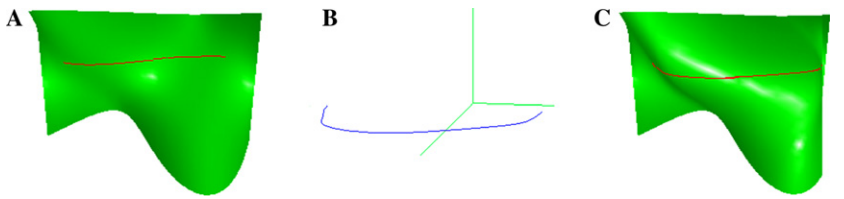

Fig. 14. Curve editing: (A) the selected source curve shown in red; (B) the destination curve shown in blue; and (C) the deformed PDE surface after curve attachment. (For interpretation of the references to colours in this figure legend, the reader is referred to the web version of this paper.)

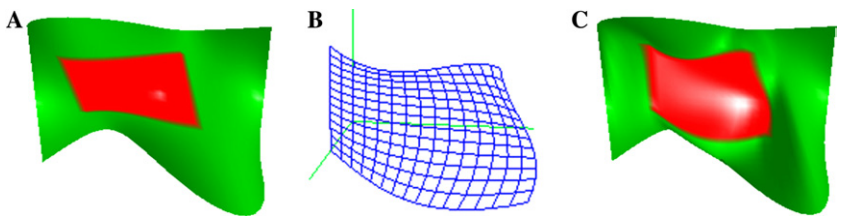

Fig. 15. Region sculpting: (A) the selected source region shown in red; (B) the B-spline destination patch; and (C) the deformed surface after area attachment. (For interpretation of the references to colours in this figure legend, the reader is referred to the web version of this paper.)

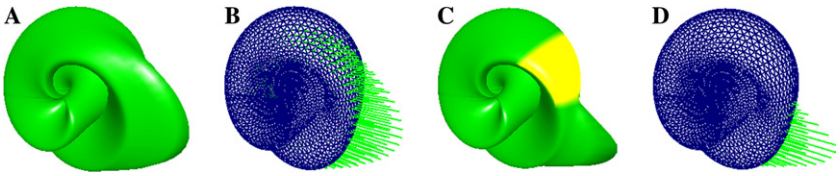

Fig. 16. Examples for point-based displacement PDE surface sculpting: (A) changing a point's location on a PDE displacement model; (B) the vector-valued displacement map on the original surface is shown in lines; (C) point sculpting outside the fixed region (yellow) of displacements; and (D) the original surface and the displacement vectors of (C). (For interpretation of the references to colours in this figure legend, the reader is referred to the web version of this paper.)

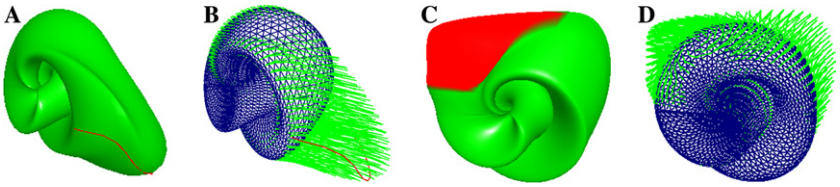

Fig. 17. Examples for curve and region editing on PDE surface displacements: (A) a PDE surface sculpted using curve editing on displacements; (B) the corresponding vector-valued displacement map on the underlying surface; (C) region manipulation of PDE displacements; (D) the original surface and the displacement vector map of (C).

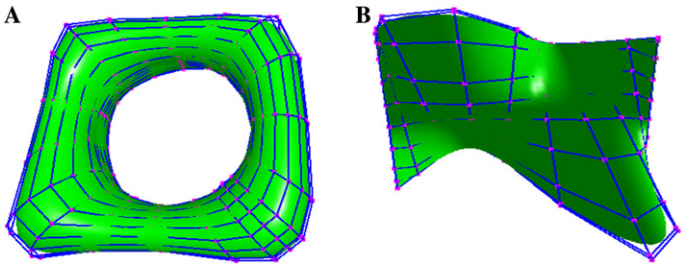

Fig. 18. B-spline approximations for PDE surfaces. The blue lines with purple points are B-spline control meshes.

5.1.3. Dynamic models

Our system supports novel physics-based PDE surface techniques including: (1) finite-difference discretization for mass-spring models; (2) multigrid-like subdivision for model refinement; and (3) finite element approximation using B-splines. Material properties and dynamic behavior can greatly enhance the interactive manipulation of conventional PDE surfaces.

5.1.4. Sculpting tools

Our system provides various manipulation toolkits to offer users the capability of interactive sculpting of the physics-based PDE surfaces/displacements. These tool-

kits include: (1) patching several PDE surfaces smoothly; (2) moving (a set of) arbitrary surface points to desired locations; (3) modifying surface normal at arbitrary data points; (4) editing surface curvatures of arbitrary surface points; (5) changing boundary conditions; (6) modifying the blending control function (i.e., $a(u,v)$) associated with the PDE; (7) specifying and enforcing a set of curve constraints; (8) deforming a set of user-specified regions to the desired shape; (9) freezing any local region(s); (10) applying local operations only on user-selected areas; (11) trimming the specified part of the surface; (12) modifying material properties such as mass, damping, and stiffness distributions locally; (13) computing the B-spline approximation of PDE surfaces; and (14) directly deforming B-spline finite elements with *forces*.

In addition to the previous PDE surface deformations through boundary and coefficient modifications, our integrated physics-based PDE modeling method offers direct and interactive surface manipulation with flexible boundary and additional constraints and physical properties which are lacking from the traditional PDE-based techniques. Our method uses an elliptic PDE to define smooth surfaces through boundary information instead of control network over the entire surface regions in spline-based techniques. Interpolating constraints can be applied naturally as generalized boundary conditions to define PDE surfaces. The sculpting of PDE surfaces through direct manipulation can be automatically obtained by the governing PDE instead of using control network of spline-based techniques to guide the surface deformation. The unified physics-based PDE surface modeling framework provides real-time manipulation of smooth surfaces with high-order continuity and material properties. It offers modeling advantages of both physics-based methods and PDE techniques.

5.2. Results and discussion

We use several numerical techniques to solve the PDE surface subject to various constraints. Table 1 summarizes the CPU time of three different numerical solvers on the PDE surface (Fig. 7B) discretized using different grids. Because the time to solve the discretized PDE mainly depends on the number of sampling grids, the time performance for other similar examples will be very close to Table 1. G-E represents Gaussian elimination method, G-S stands for Gauss–Seidel iteration, and SOR stands for SOR iteration. The iteration threshold (0.001) for the two iterative methods is the sum of the distance between sample points in successive steps. From Table

Table 1
CPU-time (seconds) of different solvers for the PDE surface (Fig. 7B) with different discretization resolutions

Grids	G-E	G-S	SOR ($\sigma = 1.25$)
15×15	0.094	1	0.593
30×30	3.187	15.016	6
60×60	122.156	64.719	34.391

1, we observe that it is generally very time consuming to solve the PDE surface of large sample grids using direct method such as Gaussian elimination. The multi-grid-like subdivision method will allow us to start from the coarse approximation of the surface at very small sample grids, then apply subdivision refinement on the coarse level to obtain a more accurate solution at finer resolution. The total CPU time of using this method is much less than directly solving the equation on the same grids.

Table 2 compares the CPU time of using Gauss–Seidel iteration and SOR iteration with the multi-grid subdivision for Fig. 9. We start solving the surface at the sample grid 15×15. The total CPU time to obtain a solution through the iterative approaches at selected grid is the sum of the numbers from the coarsest level to the current level in the same column. The choice of σ will also influence the time performance in our SOR solver. $\sigma = 1.05$, 1.15, and 1.25 for SOR1, SOR2, and SOR3, respectively.

Table 3 lists number of iterations for typical manipulation toolkits on some examples under the two iterative methods. P, N, and V represent the point, normal, and curvature manipulations, respectively. C denotes curve editing with 20 sample points, while R stands for the regional manipulation of 10×10 sample points attached to a B-spline patch. Note that, it generally takes more iterations in a coarsely sampled grid, however, the CPU time spent on the coarser grid is far less than that on the finer grid.

Besides traditional boundary conditions of PDE techniques, our system allows users to specify and enforce a large variety of additional constraints on a set of points, cross-sectional curves, and surface regions. These constraints provide more freedom to designers, making the design process of PDE surfaces more intuitive, natural, and cost-effective. We develop our prototype system using both finite-difference and B-spline finite element techniques. The advantages of these approximation techniques are that they are simple, easy to implement, and suitable for the incorporation of complicated, flexible constraints. On the other hand, the time and space complexity

Table 2
CPU-time (seconds) of different iterative methods in multigrid-like subdivision for the PDE surface (Fig. 9)

Grids	G-S	SOR1	SOR2	SOR3
15×15	1	0.109	0.437	0.593
30×30	0.266	0.079	0.171	0.282
60×60	1.1	0.156	0.344	0.5

Table 3
Number of iterations for various manipulation techniques with different sampling grids

Grids	G-S	P (Fig. 12A)	N (Fig. 12B)	V (Fig. 12C)	C (Fig. 14)	R (Fig. 15)
15×15	1438	253	146	447	497	102
30×30	1751	1420	146	494	1681	457
60×60	4000	933	146	190	3504	2000

are increased correspondingly with higher resolution as well as increased accuracy. The convergent rate of iterations depends on the initial values. Our multigrid-like subdivision method for various levels of refinements achieves an anticipated result in our experiments.

6. Conclusion

We have presented a set of interactive techniques that supports both global and local deformations of PDE-based surface models subject to general and flexible constraints. We have proposed a unified methodology that marries PDE-based parametric surface models with physics-based techniques. Physics-based modeling permits the PDE surfaces and displacements to be governed by physical laws and equipped with dynamic behavior, making PDE-based surface models more realistic and interactive than the prior kinematic PDE surfaces. Our prototype software system provides users a wide range of powerful toolkits for interactive surface and displacement sculpting including: point-based manipulation such as position modification, normal editing, and curvature control; cross-sectional design such as boundary control and the manipulation of non-isoparametric curves; and region deformations. These toolkits permit users to model and manipulate physics-based PDE surfaces and displacements intuitively and interactively in real-time. Our experiments have shown that general and flexible constraints offer users freedom and a natural interface to manipulate the physics-based PDE surface satisfying a set of design criteria and functional requirements. Our system also computes the B-spline finite element approximation of the PDE surface and allows users to interactively manipulate B-splines to support the data exchange capability in commercially available geometric modeling systems. Our unified approach and novel PDE techniques greatly expand the geometric coverage and the topological flexibility of conventional PDE surfaces, improving the utility of PDE surfaces for modeling and design applications, as well as helping the realization of the full potential of PDE technology in visual computing fields.

Acknowledgments

This research was supported in part by the NSF CAREER Award CCR-9896123, the NSF Grants DMI-9896170, IIS-0082035, IIS-0097646, and a Sloan Fellowship from Alfred P. Sloan Foundation. We would also like to thank the anonymous reviewers for their valuable suggestions that greatly improve this paper.

References

[1] W. Bohm, G. Farin, J. Kahmann, A survey of curve and surface methods in CAGD, Comput. Aided Geometric Des. 1 (1) (1984) 1–60.
[2] C. de Boor, A Practical Guide to Splines, Springer, Berlin, 1978.

[3] G. Farin, Curves and Surfaces for Computer-Aided Geometric Design, A Practical Guide, Academic Press, New York, 1996.
[4] D.R. Forsey, R.H. Bartels, Hierarchical b-spline refinement, Computer Graphics 22 (4) (1988) 205–212.
[5] L. Piegl, On NURBS: a survey, IEEE Comput. Graph. Appl. 11 (1) (1991) 55–71.
[6] L. Piegl, W. Tiller, Curve and surface constructions using rational B-splines, Comput. Aided Des. 19 (9) (1987) 485–498.
[7] R. Schneider, L. Kobbelt, Generating fair meshes with G^1 boundary conditions, in: Geometric Modeling and Processing Conference Proceedings, 2000, pp. 251–261.
[8] H.K. Zhao, S. Osher, B. Merriman, M. Kang, Implicit and non-parametric shape reconstruction from unorganized points using variational level set method, Comput. Vision Image Understand. 80 (3) (2000) 295–319.
[9] H.K. Zhao, S. Osher, R. Fedkiw, Fast surface reconstruction using level set method, in: IEEE Workshop on Variational and Level Set Methods (VLSM 01), Vancouver, Canada, 2001, pp. 194–202.
[10] D. Breen, R. Whitaker, A level-set approach for the metamorphosis of solid models, IEEE Trans. Visual. Comput. Graph. 7 (2) (2001) 173–192.
[11] M.I.G. Bloor, M.J. Wilson, Blend design as a boundary-value problem, in: I.W. Straßer (Ed.), Geometric Modeling: Theory and Practice, Springer-Verlag, Berlin, 1989, pp. 221–234.
[12] M.I.G. Bloor, M.J. Wilson, Generating blend surfaces using partial differential equations, Comput. Aided Des. 21 (3) (1989) 165–171.
[13] M.I.G. Bloor, M.J. Wilson, Using partial differential equations to generate free-form surfaces, Comput. Aided Des. 22 (4) (1990) 202–212.
[14] T.W. Lowe, M.I.G. Bloor, M.J. Wilson, Functionality in blend design, Comput. Aided Des. 22 (10) (1990) 655–665.
[15] H. Du, H. Qin, Direct manipulation and interactive sculpting of PDE surfaces, Comput. Graph. Forum 19 (3) (2000) C261–C270.
[16] H. Du, H. Qin, Dynamic PDE surfaces with flexible and general constraints, in: Proceedings of the Pacific Graphics 2000, Hong Kong, 2000, pp. 213–222.
[17] M.I.G. Bloor, M.J. Wilson, Representing PDE surfaces in terms of B-splines, Comput. Aided Des. 22 (6) (1990) 324–331.
[18] M.I.G. Bloor, M.J. Wilson, Functionality in solids obtained from partial differential equations, Comput. Suppl. 8 (1993) 21–42.
[19] M.I.G. Bloor, M.J. Wilson, Spectral approximations to PDE surfaces, Comput. Aided Des. 28 (2) (1996) 145–152.
[20] H. Ugail, M.I.G. Bloor, M.J. Wilson, Techniques for interactive design using the PDE method, ACM Trans. Graph. 18 (2) (1999) 195–212.
[21] D. Terzopoulos, J. Platt, A. Barr, K. Fleischer, Elastically deformable models, in: SIGGRAPH 87, Anaheim, CA, USA, 1987, pp. 205–214.
[22] D. Terzopoulos, K. Fleischer, Modeling inelastic deformation: viscoelasticity, plasticity, fracture, in: SIGGRAPH 88, Atlanta, GA, USA, 1988, pp. 269–278.
[23] D. Terzopoulos, K. Fleischer, Deformable models, The Visual Computer 4 (6) (1988) 306–331.
[24] G. Celniker, D. Gossard, Deformable curve and surface finite elements for free-form shape design, Comput. Graph. 25 (4) (1991) 165–170.
[25] H. Qin, D. Terzopoulos, D-NURBS: A physics-based framework for geometric design, IEEE Trans. Visual. Comput. Graph. 2 (1) (1996) 85–96.
[26] D. Terzopoulos, H. Qin, Dynamic NURBS with geometric constraints for interactive sculpting, ACM Trans. Graph. 13 (2) (1994) 103–136.
[27] X. Ye, T.R. Jackson, N.M. Patrikalakis, Geometric design of functional surfaces, Comput. Aided Des. 28 (9) (1996) 741–752.
[28] F. Dachille, H. Qin, A. Kaufman, J. El-Sana, Haptic sculpting of dynamic surfaces, in: Computer Graphics (1999 SIGGRAPH Symposium on Interactive 3D Graphics), Atlanta, Georgia, 1999, pp. 103–110.
[29] D. Baraff, A. Witkin, Large steps in cloth simulation, in: SIGGRAPH 98, Orlando, FL, USA, 1998, pp. 43–54.

[30] M. Carignan, Y. Yang, N.M. Thalmann, D. Thalmann, Dressing animated synthetic actors with complex deformable clothes, in: SIGGRAPH 92, Chicago, IL, USA, 1992, pp. 99–104.
[31] J. Hua, H. Qin, Dynamic implicit solids with constraints for haptic sculpting, in: Proc. Internat. Conf. on Shape Modeling and Applications, Banff, Alberta, Canada, 2002, pp. 119–128.
[32] R.M. Koch, M.H. Gross, F.R. Carls, D.F. von Büren, G. Fankhauser, Y.I.H. Parish, Simulating facial surgery using finite element models, in: SIGGRAPH 1996, USA, 1996, pp. 421–428.
[33] C. Mandal, H. Qin, B.C. Vemuri, A novel FEM-based dynamic framework for subdivision surfaces, in: Fifth Symposium on Solid Modeling, Ann Arbor, MI, USA, 1999, pp. 191–202.
[34] K.T. McDonnell, H. Qin, R.A. Wlodarczyk, Virtual clay: a real-time sculpting system with haptic toolkits, in: Proceedings of 2001 ACM Symposium on Interactive 3D Graphics, Salt Lake City, UT, USA, 2001, pp. 179–190.
[35] K. Singh, E. Fiume, Wires: a geometric deformation technique, in: SIGGRAPH 1998, Orlando, FL, USA, 1998, pp. 405–414.
[36] H. Biermann, A. Levin, D. Zorin, Piecewise smooth subdivision surfaces with normal control, in: SIGGRAPH 2000, New Orleans, LA, USA, 2000, pp. 113–120.
[37] R. Cook, Shade trees, Computer Graphics (Proceeding of SIGGRAPH 1984) 18 (3) (1984) 223–231.
[38] V. Krishnamurthy, M. Levoy, Fitting smooth surface to dense polygon meshes, in: SIGGRAPH 1996, USA, 1996, pp. 313–324.
[39] A. Lee, H. Moreton, H. Hoppe, Displaced subdivision surfaces, in: SIGGRAPH 2000, New Orleans, LA, USA, 2000, pp. 85–94.
[40] L. Kobbelt, T. Bareuther, H.-P. Seidel, Multiresolution shape deformations for meshes with dynamic vertex connectivity, in: Proceedings of the EuroGraphics 2000, Interlaken, Switzerland, 2000, pp. C417–C427.
[41] R. Jagnow, J. Dorsey, Virtual sculpting with haptic displacement maps, in: Proceedings of the Graphics Interface, Calgary, Alberta, 2002, pp. 125–132.
[42] G. Strang, Introduction to Applied Mathematics, Wellesley-Cambridge Press, 1986.
[43] N. Dyn, D. Levin, J.A. Gregory, A butterfly subdivision scheme for surface interpolation with tension control, ACM Transaction on Graphics 9 (2) (1990) 160–169.
[44] L. Kobbelt, Interpolary subdivision on open quadrilateral nets with arbitrary topology, Computer Graphics Forum 15 (3) (1996) 409–420.